G54FOP: Lecture 14

The Polymorphic Lambda Calculus (System F)

Henrik Nilsson

University of Nottingham, UK

This Lecture

- Limitations of the simply typed λ-calculus.
- The polymorphic lambda calculus (System F)
- Examples illustrating the power of system F

Rcp: The Simply Typed λ-Calculus (1)

T	$\rightarrow$		*types:*
	$\mid$	B	*fixed set of base types*
	$\mid$	$T \rightarrow T$	*type of functions*
Γ	$\rightarrow$		*contexts:*
	$\mid$	$\emptyset$	*empty context*
	$\mid$	$\Gamma, x : T$	*context extension*

Note: Need at least ***one*** base type, or there is no way to construct a type of finite size.

Rcp: The Simply Typed λ-Calculus (2)

t	$\rightarrow$		*terms:*
	$\mid$	x	*variable*
	$\mid$	c	*constant (optional)*
	$\mid$	$\lambda x : T \,.\, t$	*abstraction*
	$\mid$	$t\ t$	*application*
v	$\rightarrow$		*values:*
	$\mid$	c	*constant (optional)*
	$\mid$	$\lambda x : T \,.\, t$	*abstraction*

Rcp: The Simply Typed λ-Calculus (3)

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \quad \text{(T-VAR)}$$

$$\frac{c \text{ is a constant of type } T}{\Gamma \vdash c : T} \quad \text{(T-CONST-c)}$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1 \,.\, t_2 : T_1 \rightarrow T_2} \quad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \quad \text{(T-APP)}$$

Example: TWICE (1)

Consider defining a function twice:

$$\mathrm{twice}(f, x) \;=\; f(f(x))$$

Easy in the untyped λ-calculus:

$$\mathtt{TWICE} \;\equiv\; \lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\ (\mathtt{f}\ \mathtt{x})$$

What about the ***simply typed*** λ-calculus?

$$\mathtt{TWICE} \;\equiv\; \lambda \mathtt{f}{:}???.\lambda \mathtt{x}{:}???.\mathtt{f}\ (\mathtt{f}\ \mathtt{x})$$

What should the types of the arguments be?
Can `TWICE` be used for, say, both `Bool` and `Nat`?

Example: TWICE (2)

Suppose `Bool`, `Nat` $\in B$.

What matters is that the types would be different even if we were to encode them in the base calculus.

Thus we need a ***separate*** definition for ***each*** type at which we want to use `TWICE`:

$$\mathtt{TWICEBOOL} \;\equiv\; \lambda \mathtt{f}{:}\mathtt{Bool} \rightarrow \mathtt{Bool}.\lambda \mathtt{x}{:}\mathtt{Bool}.\mathtt{f}\ (\mathtt{f}\ \mathtt{x})$$

$$\mathtt{TWICENAT} \;\equiv\; \lambda \mathtt{f}{:}\mathtt{Nat} \rightarrow \mathtt{Nat}.\lambda \mathtt{x}{:}\mathtt{Nat}.\mathtt{f}\ (\mathtt{f}\ \mathtt{x})$$

$$\mathtt{TWICENATFUN} \;\equiv\; \lambda \mathtt{f}{:}(\mathtt{Nat} \rightarrow \mathtt{Nat}) \rightarrow (\mathtt{Nat} \rightarrow \mathtt{Nat}).\ \lambda \mathtt{x}{:}\mathtt{Nat} \rightarrow \mathtt{Nat}.\mathtt{f}\ (\mathtt{f}\ \mathtt{x})$$

Example: TWICE (3)

We have been forced to define ***essentially the same*** function over and over.

Common CS sensibility suggests ***abstraction*** over the ***varying*** part; i.e., here ***the type***!

Thus, we would like to do something like:

$$\mathtt{TWICEPOLY} \;\equiv\; \Lambda \mathtt{T}.\lambda \mathtt{f}{:}\mathtt{T} \rightarrow \mathtt{T}.\lambda \mathtt{x}{:}\mathtt{T}.\mathtt{f}\ (\mathtt{f}\ \mathtt{x})$$

Now:

$$\mathtt{TWICEBOOL} \;\equiv\; \mathtt{TWICEPOLY}\ [\mathtt{Bool}]$$

$$\mathtt{TWICENAT} \;\equiv\; \mathtt{TWICEPOLY}\ [\mathtt{Nat}]$$

$$\mathtt{TWICENATFUN} \;\equiv\; \mathtt{TWICEPOLY}\ [\mathtt{Nat} \rightarrow \mathtt{Nat}]$$

System F: Abstract Syntax (1)

T	$\rightarrow$		*types:*
	$\mid$	$B \mid T \rightarrow T$	*[as for simply typed]*
	$\mid$	$\forall X \,.\, T$	*universally quantified type*
Γ	$\rightarrow$		*contexts:*
	$\mid$	$\emptyset \mid \Gamma, x : T$	*[as for simply typed]*
	$\mid$	Γ, X	*extension with type variable*

System F: Abstract Syntax (2)

$t \rightarrow$ *terms:*

$\mid\ x \mid c \mid \lambda x{:}T\,.\,t \mid t\ t$ *[as for simply typed]*

$\mid\ \Lambda X\,.\,t$ *type abstraction*

$\mid\ t\ [T]$ *type application*

$v \rightarrow$ *values:*

$\mid\ c \mid \lambda x{:}T\,.\,t$ *[as for simply typed]*

$\mid\ \Lambda X\,.\,t$ *type abstraction value*

System F: Typing Rules

T-VAR, (T-CONST-c), T-ABS, T-APP are as before (omitted).

Additional typing rules:

$$\frac{\Gamma, X \vdash t : T}{\Gamma \vdash \Lambda X\,.\,t : \forall X\,.\,T} \quad \text{(T-TABS)}$$

$$\frac{\Gamma \vdash t_1 : \forall X\,.\,T_{12}}{\Gamma \vdash t_1\ [T_2] : [X \mapsto T_2]\ T_{12}} \quad \text{(T-TAPP)}$$

System F: Evaluation Rules

E-APP1, E-APP2, E-APPABS are as before:

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{(E-APP1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \quad \text{(E-APP2)}$$

$$(\lambda x{:}T_{11}\,.\,t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \quad \text{(E-APPABS)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ [T_2] \longrightarrow t_1'\ [T_2]} \quad \text{(E-TAPP)}$$

$$(\Lambda X\,.\,t_{12})\ [T_2] \longrightarrow [X \mapsto T_2]\ t_{12} \quad \text{(E-TAPPABS)}$$

Exercise

Given

$$\mathtt{ID} \equiv \Lambda\mathtt{T}.\lambda\mathtt{x{:}T.x}$$
$$\Gamma_1 = \emptyset, \mathtt{Nat}, \mathtt{5} : \mathtt{Nat}$$

type check $\mathtt{ID}\ [\mathtt{Nat}]\ \mathtt{5}$ in context Γ_1.

(On whiteboard)

System F: Church Booleans (1)

Recall untyped encoding:

$$\mathtt{TRUE} \equiv \lambda\mathtt{t}.\lambda\mathtt{f.t}$$
$$\mathtt{FALSE} \equiv \lambda\mathtt{t}.\lambda\mathtt{f.f}$$

We need to:

- assign a ***common*** type to these two terms;
- need to work for ***arbitrary*** argument types.

Parametrise on the type:

$$\mathtt{CBOOL} \equiv \forall\mathtt{X.X}\rightarrow\mathtt{X}\rightarrow\mathtt{X}$$

System F: Church Booleans (2)

$\mathtt{CBOOL} \equiv \forall\mathtt{X.X}\rightarrow\mathtt{X}\rightarrow\mathtt{X}$

$\mathtt{TRUE} : \mathtt{CBOOL}$

$\mathtt{TRUE} \equiv \Lambda\mathtt{X}.\lambda\mathtt{t{:}X}.\lambda\mathtt{f{:}X.t}$

$\mathtt{FALSE} : \mathtt{CBOOL}$

$\mathtt{FALSE} \equiv \Lambda\mathtt{X}.\lambda\mathtt{t{:}X}.\lambda\mathtt{f{:}X.f}$

$\mathtt{NOT} : \mathtt{CBOOL}\rightarrow\mathtt{CBOOL}$

$\mathtt{NOT} \equiv \lambda\mathtt{b{:}CBOOL}.\Lambda\mathtt{X}.\lambda\mathtt{t{:}X}.\lambda\mathtt{f{:}X.b}\ [\mathtt{X}]\ \mathtt{f}\ \mathtt{t}$

Normalization

System F is strongly normalizing, like the simply typed λ-calculus.

Homework

- Given $\mathtt{1} : \mathtt{Nat}$ and $\mathtt{2} : \mathtt{Nat}$, write down a type-correct application of $\mathtt{TRUE}$ to $\mathtt{1}$ and $\mathtt{2}$ such that the result is $\mathtt{1}$.
- Evaluate the above term using the evaluation rules.
- Prove $\mathtt{TRUE} : \mathtt{CBOOL}$.
- Prove $\mathtt{NOT} : \mathtt{CBOOL}\rightarrow\mathtt{CBOOL}$.
- Provide a suitable definition of logical conjunction, $\mathtt{AND}$.